FIG. 1

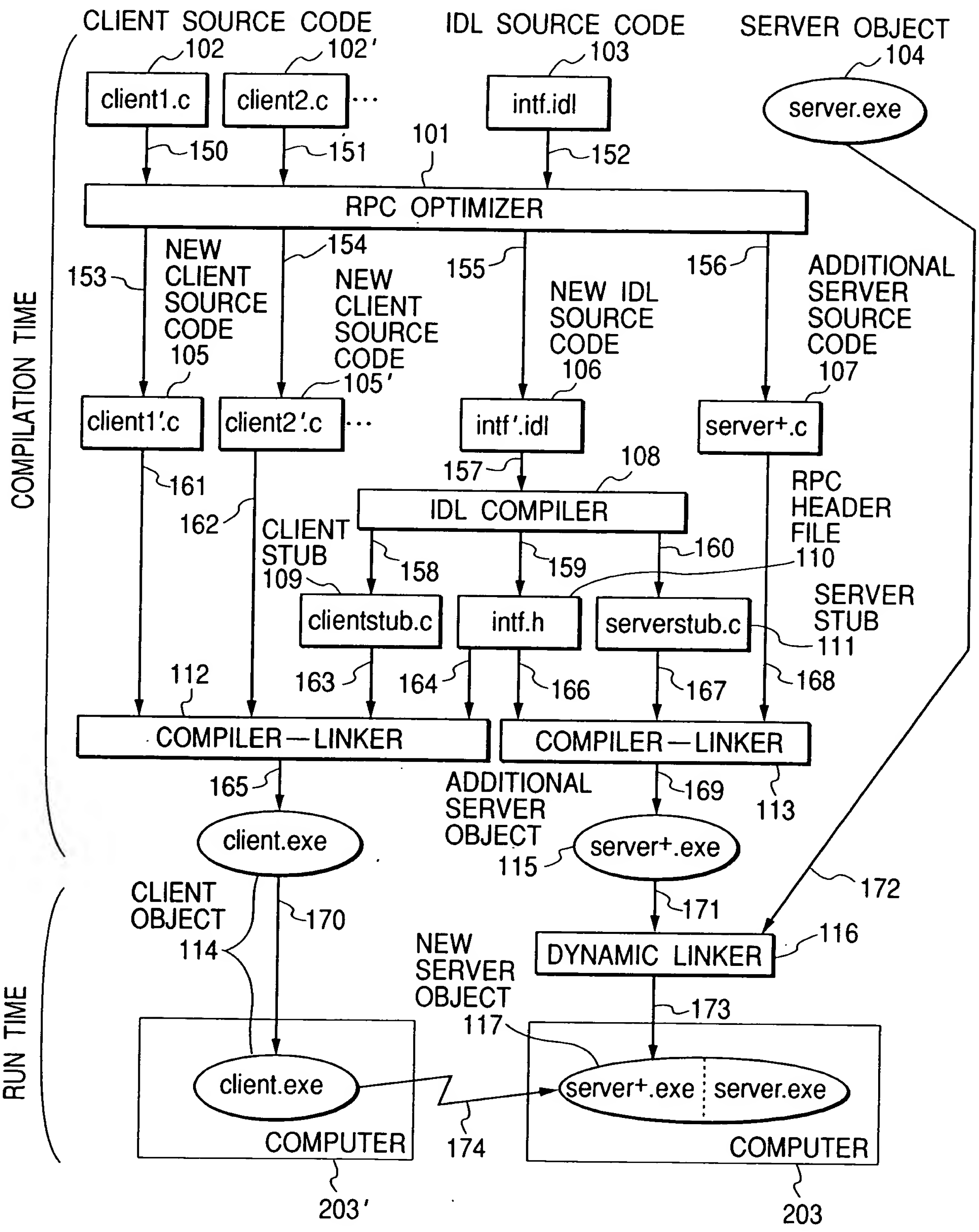

CLIENT SOURCE CODE
102
102'
client1.c
client2.c
IDL SOURCE CODE
103
intf.idl
SERVER OBJECT
104
server.exe
150
151
101
152
RPC OPTIMIZER
153
NEW CLIENT SOURCE CODE 105
154
NEW CLIENT SOURCE CODE 105'
155
NEW IDL SOURCE CODE 106
156
ADDITIONAL SERVER SOURCE CODE 107
COMPILATION TIME
client1'.c
client2'.c
intf'.idl
server+.c
161
162
157
108
IDL COMPILER
RPC HEADER FILE 110
CLIENT STUB 109
158
159
160
clientstub.c
intf.h
serverstub.c
SERVER STUB 111
112
163
164
166
167
168
COMPILER—LINKER
COMPILER—LINKER
165
ADDITIONAL SERVER OBJECT 115
169
113
client.exe
server+.exe
172
CLIENT OBJECT 114
170
171
NEW SERVER OBJECT 117
DYNAMIC LINKER
116
173
RUN TIME
client.exe
server+.exe
server.exe
COMPUTER
COMPUTER
174
203'
203

FIG. 2

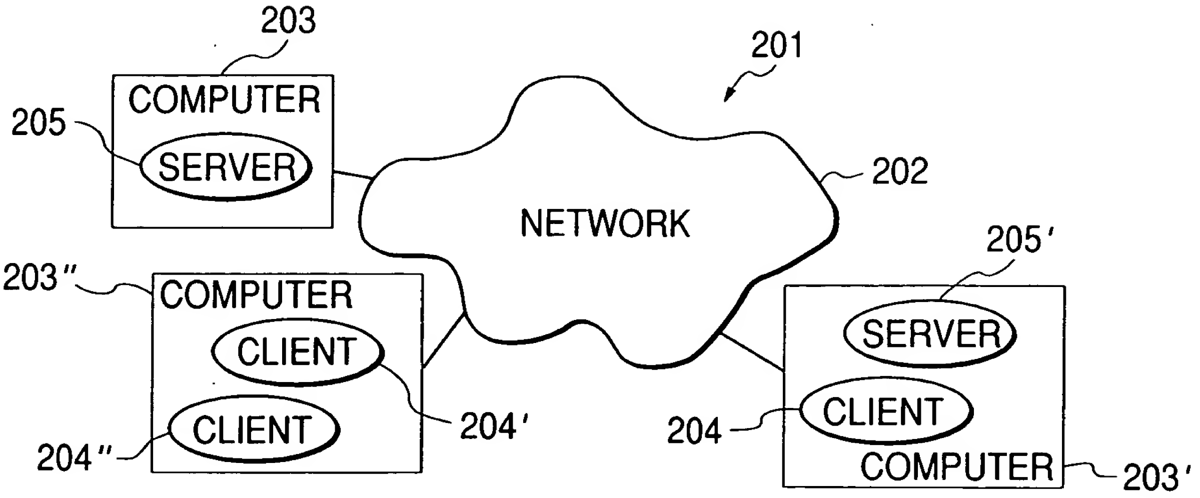

203
201
COMPUTER
205
SERVER
202
NETWORK
205'
203"
COMPUTER
CLIENT
SERVER
CLIENT
204'
204
CLIENT
204"
CLIENT
COMPUTER
203'

FIG. 3

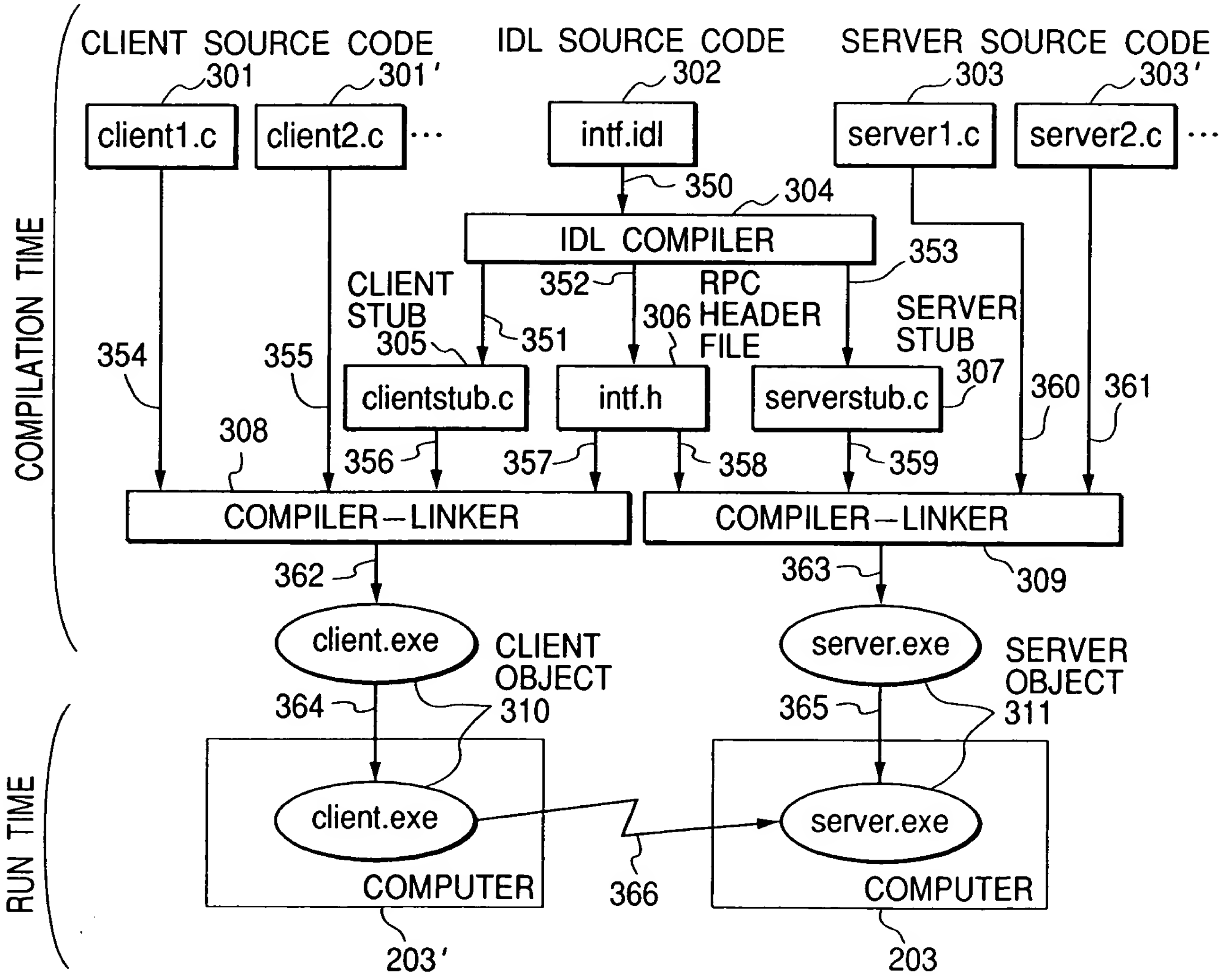

CLIENT SOURCE CODE
IDL SOURCE CODE
SERVER SOURCE CODE
301
301'
302
303
303'
client1.c
client2.c
...
intf.idl
server1.c
server2.c
...
350
304
IDL COMPILER
353
COMPILATION TIME
352
CLIENT STUB
305
351
306
RPC HEADER FILE
SERVER STUB
307
354
355
clientstub.c
intf.h
serverstub.c
360
361
308
356
357
358
359
COMPILER–LINKER
COMPILER–LINKER
362
363
309
client.exe
CLIENT OBJECT
310
server.exe
SERVER OBJECT
311
364
365
RUN TIME
client.exe
server.exe
COMPUTER
366
COMPUTER
203'
203

FIG. 4

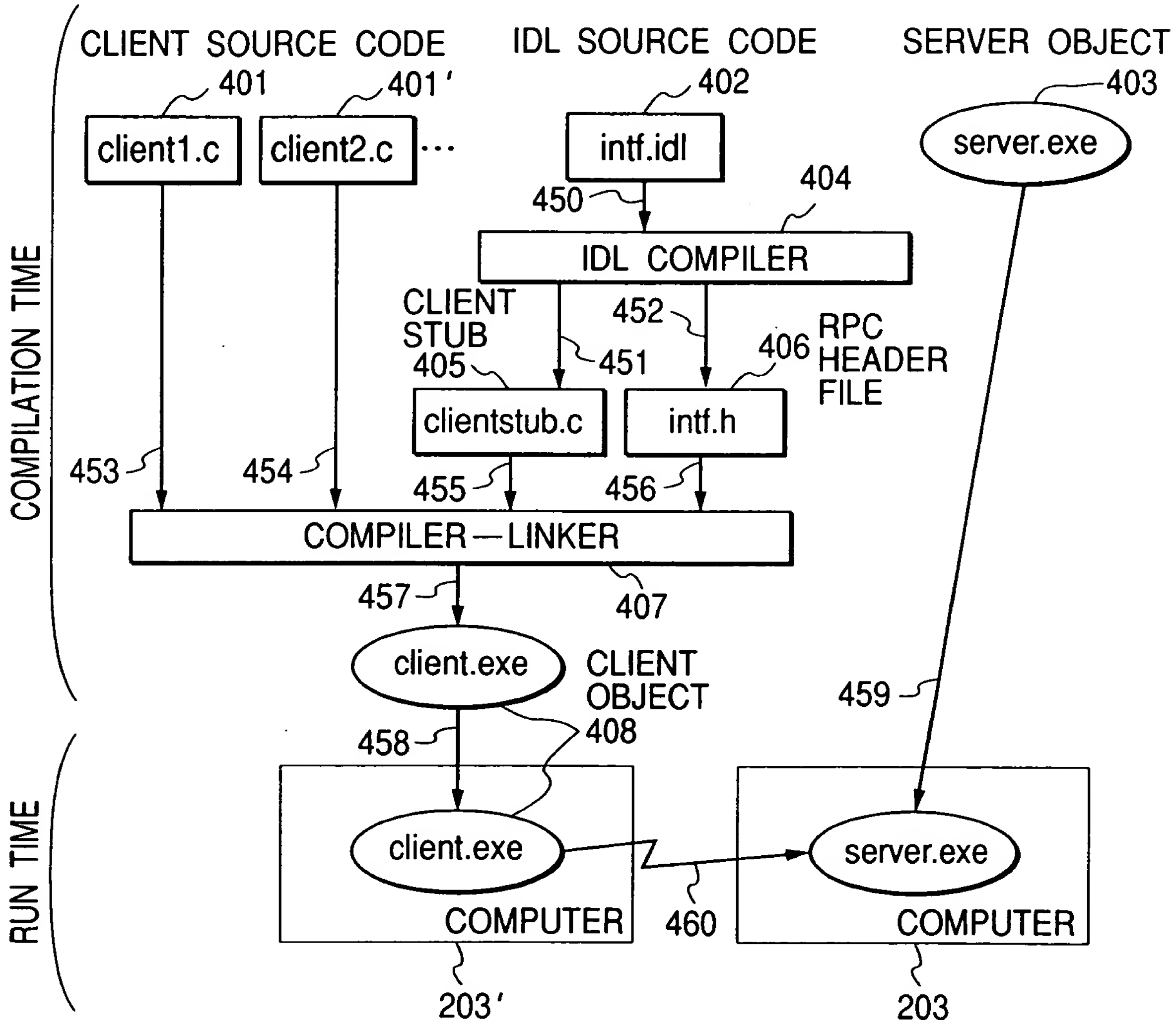

CLIENT SOURCE CODE
IDL SOURCE CODE
SERVER OBJECT
401
401'
402
403
client1.c
client2.c
...
intf.idl
server.exe
450
404
IDL COMPILER
CLIENT STUB
405
452
451
406
RPC HEADER FILE
clientstub.c
intf.h
COMPILATION TIME
453
454
455
456
COMPILER—LINKER
457
407
client.exe
CLIENT OBJECT
408
458
459
RUN TIME
client.exe
server.exe
COMPUTER
460
COMPUTER
203'
203

FIG. 5

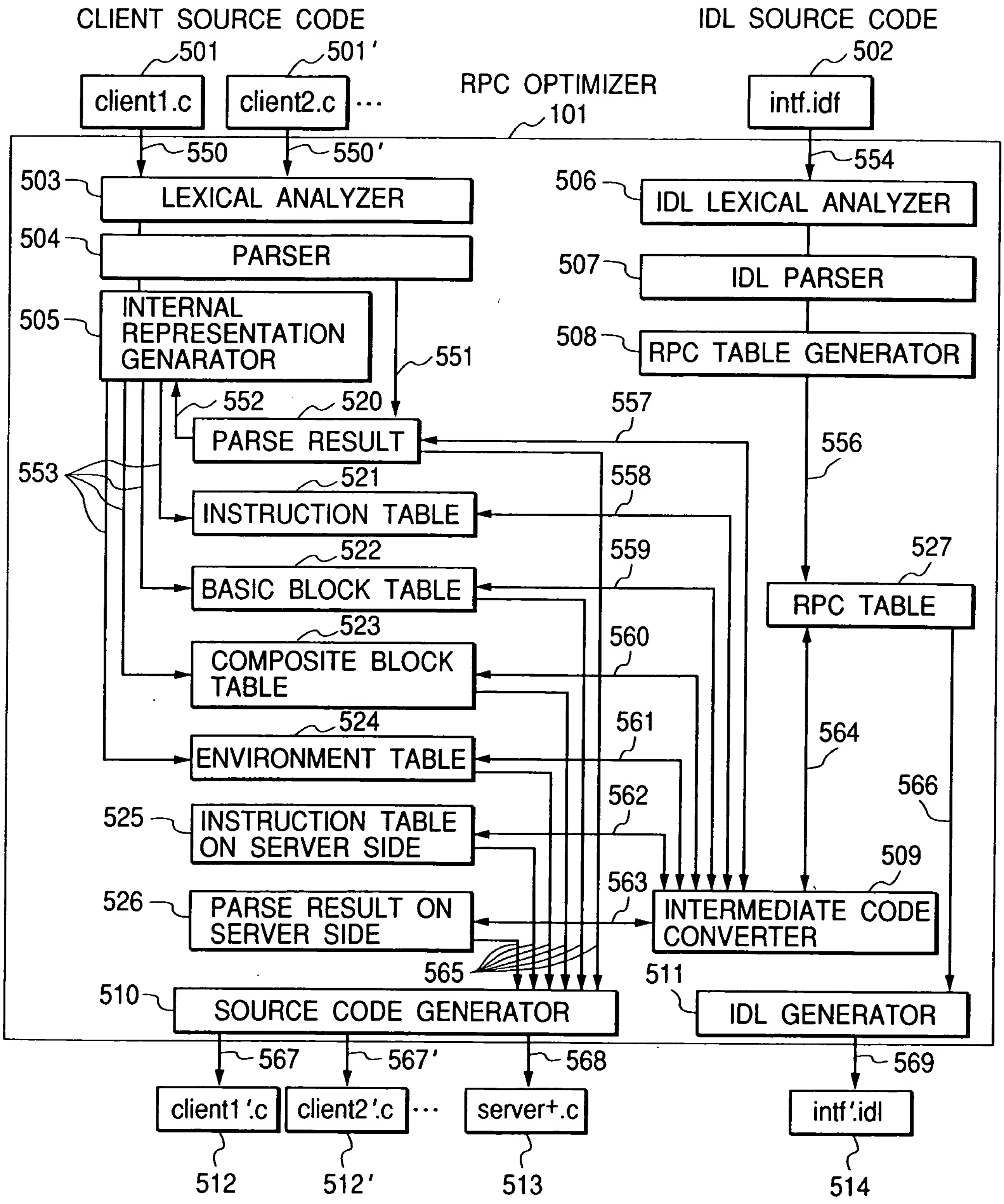

CLIENT SOURCE CODE
IDL SOURCE CODE
501
501'
502
client1.c
client2.c ···
RPC OPTIMIZER
101
intf.idf
550
550'
554
503
LEXICAL ANALYZER
506
IDL LEXICAL ANALYZER
504
PARSER
507
IDL PARSER
505
INTERNAL REPRESENTATION GENARATOR
551
508
RPC TABLE GENERATOR
552
520
557
PARSE RESULT
556
553
521
558
INSTRUCTION TABLE
522
559
527
BASIC BLOCK TABLE
RPC TABLE
523
560
COMPOSITE BLOCK TABLE
524
561
564
ENVIRONMENT TABLE
566
562
525
INSTRUCTION TABLE ON SERVER SIDE
509
563
526
PARSE RESULT ON SERVER SIDE
INTERMEDIATE CODE CONVERTER
565
511
510
SOURCE CODE GENERATOR
IDL GENERATOR
567
567'
568
569
client1'.c
client2'.c ···
server+.c
intf'.idl
512
512'
513
514

FIG. 6

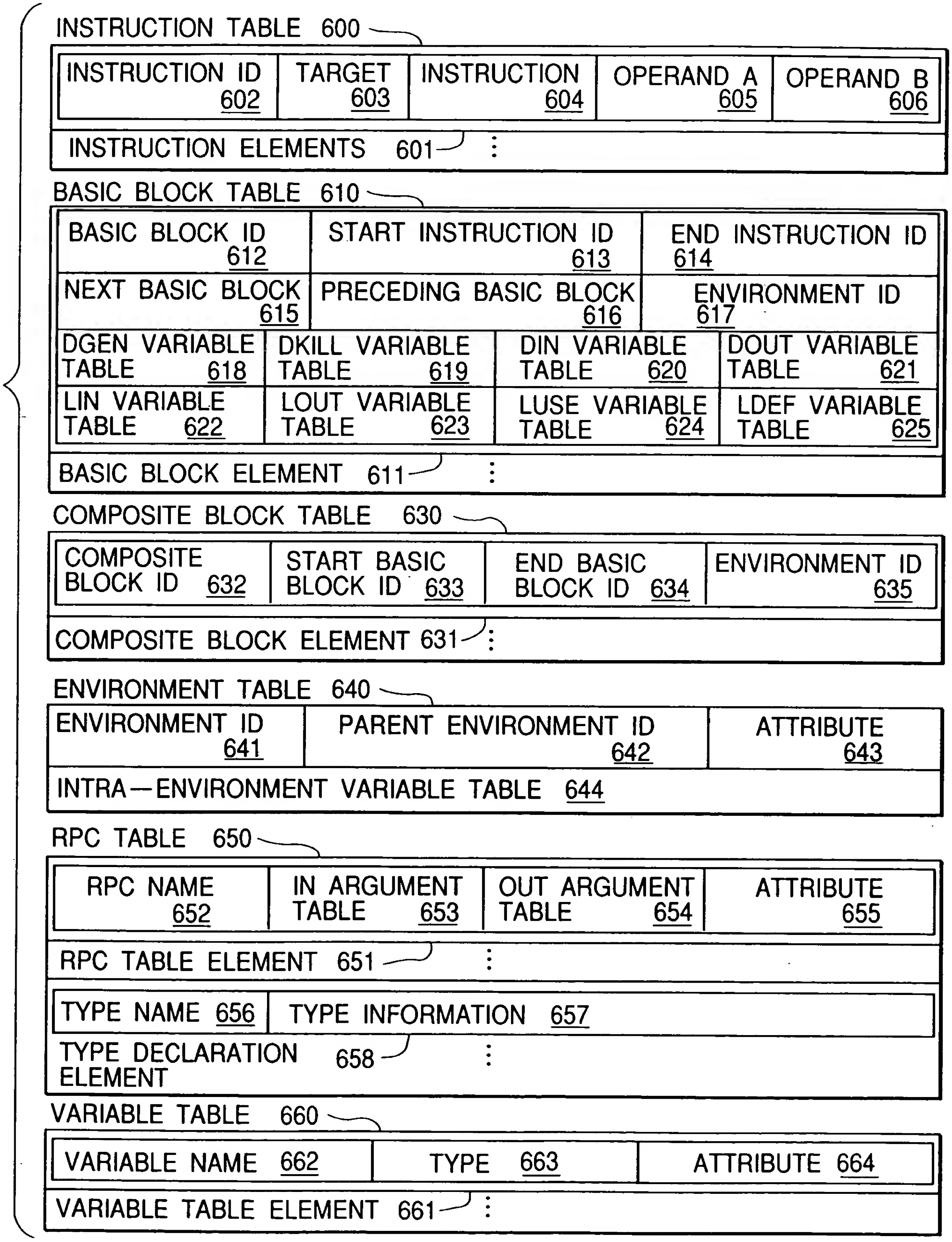
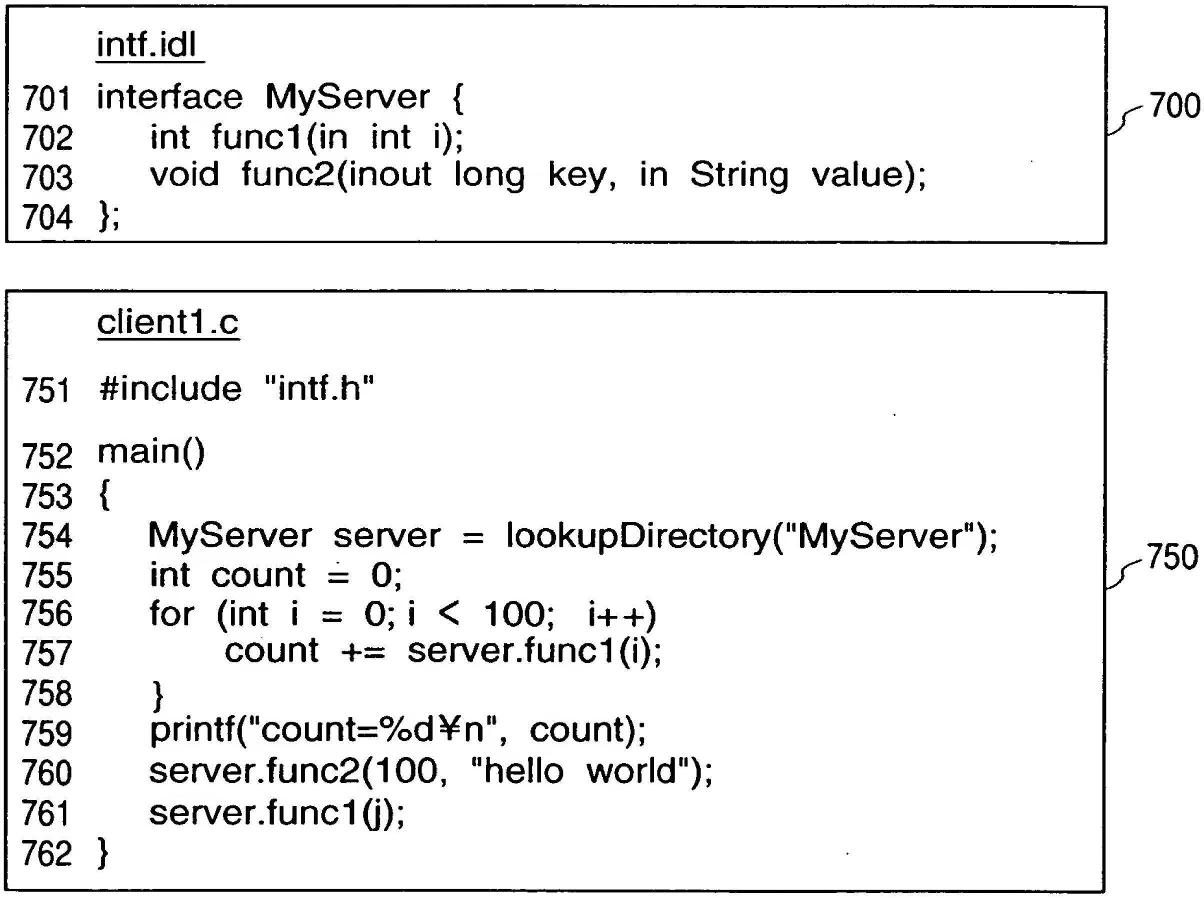

INSTRUCTION TABLE 600
INSTRUCTION ID 602
TARGET 603
INSTRUCTION 604
OPERAND A 605
OPERAND B 606
INSTRUCTION ELEMENTS 601
BASIC BLOCK TABLE 610
BASIC BLOCK ID 612
START INSTRUCTION ID 613
END INSTRUCTION ID 614
NEXT BASIC BLOCK 615
PRECEDING BASIC BLOCK 616
ENVIRONMENT ID 617
DGEN VARIABLE TABLE 618
DKILL VARIABLE TABLE 619
DIN VARIABLE TABLE 620
DOUT VARIABLE TABLE 621
LIN VARIABLE TABLE 622
LOUT VARIABLE TABLE 623
LUSE VARIABLE TABLE 624
LDEF VARIABLE TABLE 625
BASIC BLOCK ELEMENT 611
COMPOSITE BLOCK TABLE 630
COMPOSITE BLOCK ID 632
START BASIC BLOCK ID 633
END BASIC BLOCK ID 634
ENVIRONMENT ID 635
COMPOSITE BLOCK ELEMENT 631
ENVIRONMENT TABLE 640
ENVIRONMENT ID 641
PARENT ENVIRONMENT ID 642
ATTRIBUTE 643
INTRA—ENVIRONMENT VARIABLE TABLE 644
RPC TABLE 650
RPC NAME 652
IN ARGUMENT TABLE 653
OUT ARGUMENT TABLE 654
ATTRIBUTE 655
RPC TABLE ELEMENT 651
TYPE NAME 656
TYPE INFORMATION 657
TYPE DECLARATION ELEMENT 658
VARIABLE TABLE 660
VARIABLE NAME 662
TYPE 663
ATTRIBUTE 664
VARIABLE TABLE ELEMENT 661

FIG. 7



intf.idl

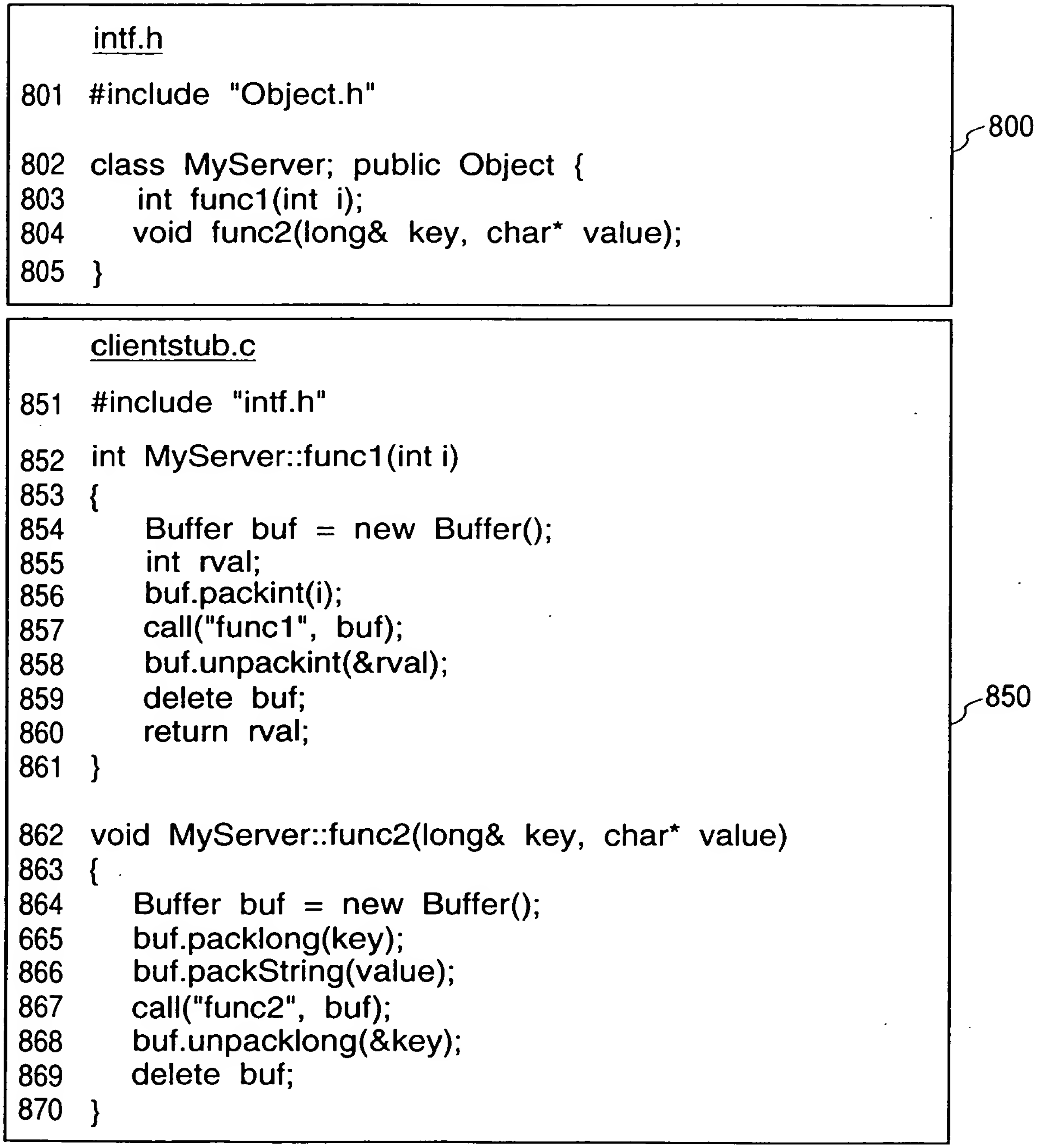
```
interface MyServer {
    int func1(in int i);
    void func2(inout long key, in String value);
};
```

client1.c

```
#include "intf.h"

main()
{
    MyServer server = lookupDirectory("MyServer");
    int count = 0;
    for (int i = 0; i < 100;  i++)
          count += server.func1(i);
    }
    printf("count=%d¥n", count);
    server.func2(100, "hello world");
    server.func1(j);
}
```

FIG. 8



intf.h

```
#include "Object.h"

class MyServer; public Object {
    int func1(int i);
    void func2(long& key, char* value);
}
```

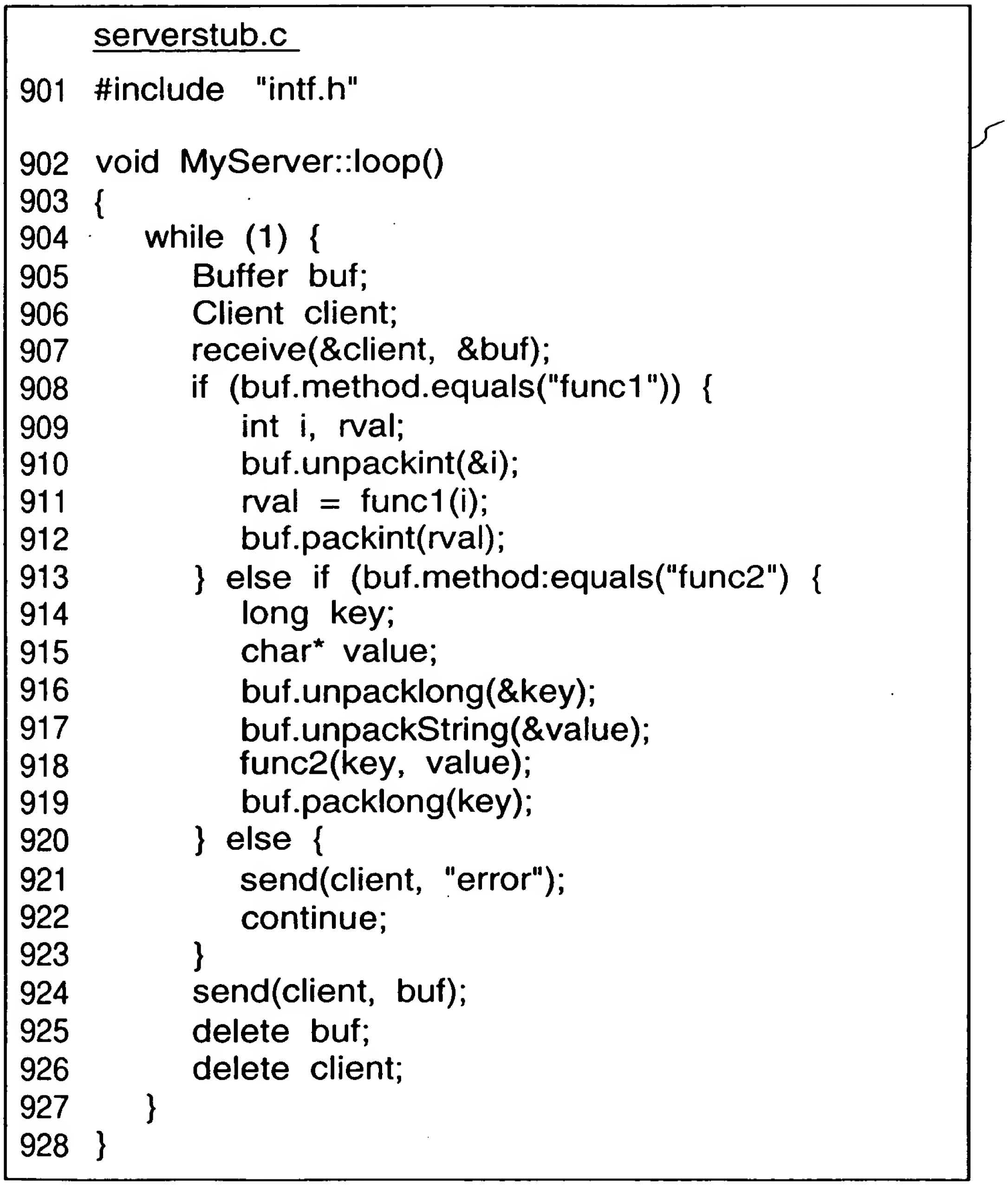
clientstub.c

```
#include "intf.h"

int MyServer::func1(int i)
{
     Buffer buf = new Buffer();
     int rval;
     buf.packint(i);
     call("func1", buf);
     buf.unpackint(&rval);
     delete buf;
     return rval;
}

void MyServer::func2(long& key, char* value)
{
    Buffer buf = new Buffer();
    buf.packlong(key);
    buf.packString(value);
    call("func2", buf);
    buf.unpacklong(&key);
    delete buf;
}
```

FIG. 9



serverstub.c

```
#include  "intf.h"

void MyServer::loop()
{
    while (1) {
        Buffer buf;
        Client client;
        receive(&client, &buf);
        if (buf.method.equals("func1")) {
            int i, rval;
            buf.unpackint(&i);
            rval = func1(i);
            buf.packint(rval);
        } else if (buf.method:equals("func2") {
            long key;
            char* value;
            buf.unpacklong(&key);
            buf.unpackString(&value);
            func2(key, value);
            buf.packlong(key);
        } else {
            send(client, "error");
            continue;
        }
        send(client, buf);
        delete buf;
        delete client;
    }
}
```

FIG. 10

intf'.idl

```
1001 interface MyServer {
1002     int func1(in int i);
1003     void func2(inout long key, in String value);
1004     void func3(inout int count);
1005     void func4(in int i);
1006 };
```

1000

client1'.c

```
1011 #include "intf'.h"

1012 main()
1013 {
1014     MyServer server = lookupDirectory("MyServer");
1015     int count = 0;
1016     server.func3(count);
1017     printf("count=%d¥n", count);
1018     server.func4(j);
1019 }
```

1010

server+.c

```
1031 #include  "intf'.h"

1032 void MyServer::func3(int& count)
1033 {
1034     for (int i = 0; i < 100; i++)
1035         count += server.func1(i);
1036 }

1037 void MyServer::func4(lint count)
1038 {
1039     server.func2(100,"hello world");
1040     server.func1(count);
1041 }
```

1030

FIG. 11

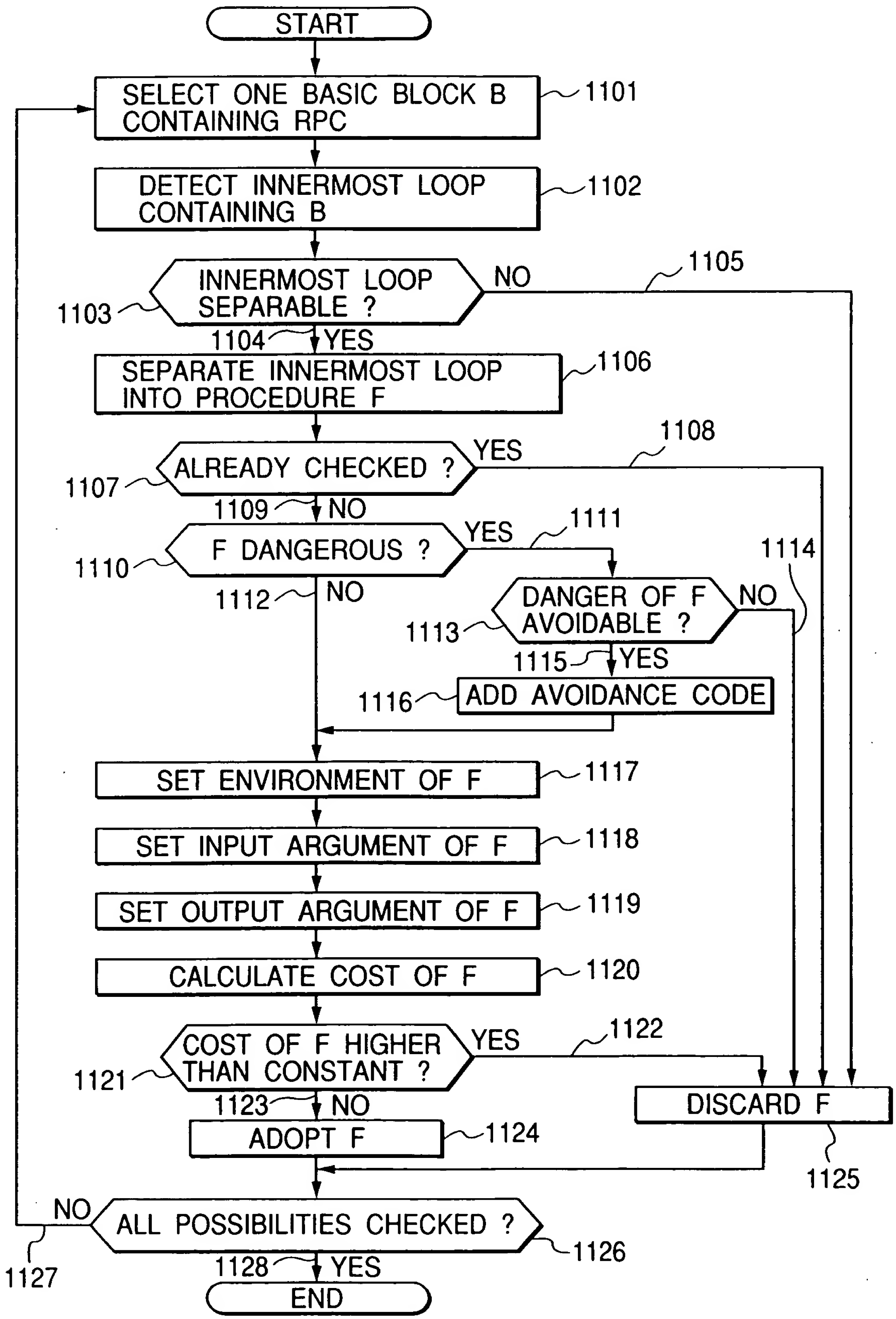

START
SELECT ONE BASIC BLOCK B CONTAINING RPC
1101
DETECT INNERMOST LOOP CONTAINING B
1102
INNERMOST LOOP SEPARABLE ?
1103
NO
1105
1104
YES
SEPARATE INNERMOST LOOP INTO PROCEDURE F
1106
ALREADY CHECKED ?
1107
YES
1108
1109
NO
F DANGEROUS ?
1110
YES
1111
1112
NO
DANGER OF F AVOIDABLE ?
1113
NO
1114
1115
YES
ADD AVOIDANCE CODE
1116
SET ENVIRONMENT OF F
1117
SET INPUT ARGUMENT OF F
1118
SET OUTPUT ARGUMENT OF F
1119
CALCULATE COST OF F
1120
COST OF F HIGHER THAN CONSTANT ?
1121
YES
1122
1123
NO
ADOPT F
1124
DISCARD F
1125
NO
ALL POSSIBILITIES CHECKED ?
1126
1127
1128
YES
END

FIG. 12

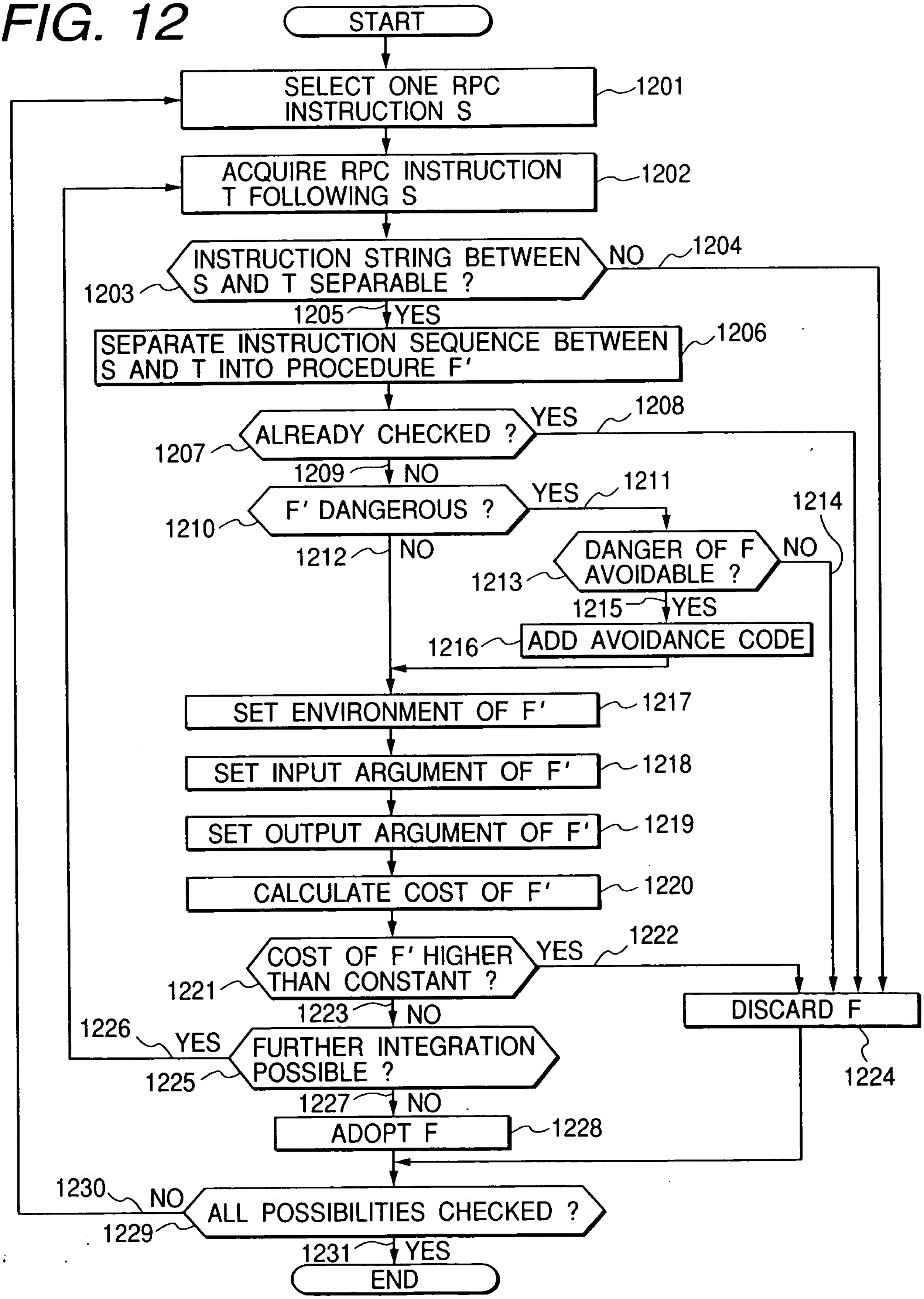

START
SELECT ONE RPC INSTRUCTION S
1201
ACQUIRE RPC INSTRUCTION T FOLLOWING S
1202
INSTRUCTION STRING BETWEEN S AND T SEPARABLE ?
1203
NO
1204
1205
YES
SEPARATE INSTRUCTION SEQUENCE BETWEEN S AND T INTO PROCEDURE F'
1206
ALREADY CHECKED ?
1207
YES
1208
1209
NO
F' DANGEROUS ?
1210
YES
1211
1214
1212
NO
DANGER OF F AVOIDABLE ?
1213
NO
1215
YES
ADD AVOIDANCE CODE
1216
SET ENVIRONMENT OF F'
1217
SET INPUT ARGUMENT OF F'
1218
SET OUTPUT ARGUMENT OF F'
1219
CALCULATE COST OF F'
1220
COST OF F' HIGHER THAN CONSTANT ?
1221
YES
1222
DISCARD F
1224
1223
NO
1226
YES
FURTHER INTEGRATION POSSIBLE ?
1225
1227
NO
ADOPT F
1228
1230
NO
ALL POSSIBILITIES CHECKED ?
1229
1231
YES
END

FIG. 13

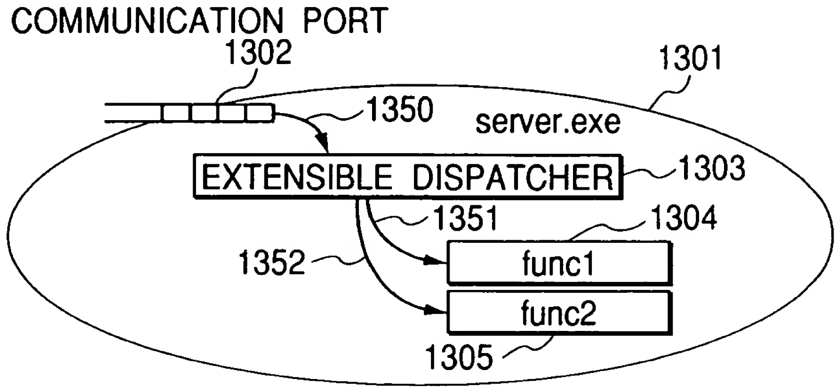

COMMUNICATION PORT
1302
1301
1350
server.exe
EXTENSIBLE DISPATCHER
1303
1351
1304
1352
func1
func2
1305

FIG. 14

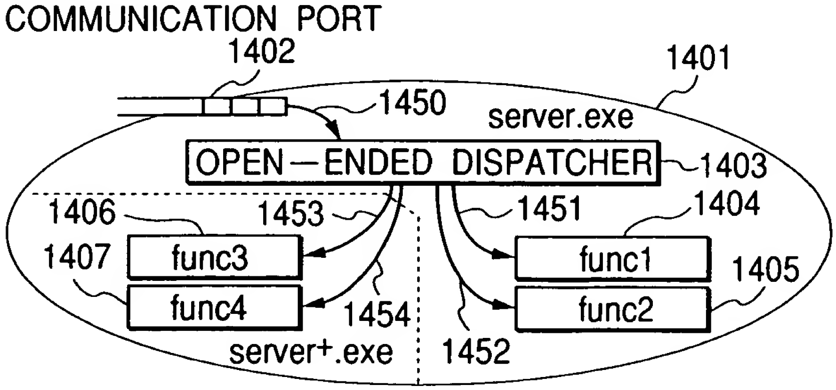

COMMUNICATION PORT
1402
1401
1450
server.exe
OPEN—ENDED DISPATCHER
1403
1406
1453
1451
1404
1407
func3
func1
1405
func4
func2
1454
server+.exe
1452

FIG. 15

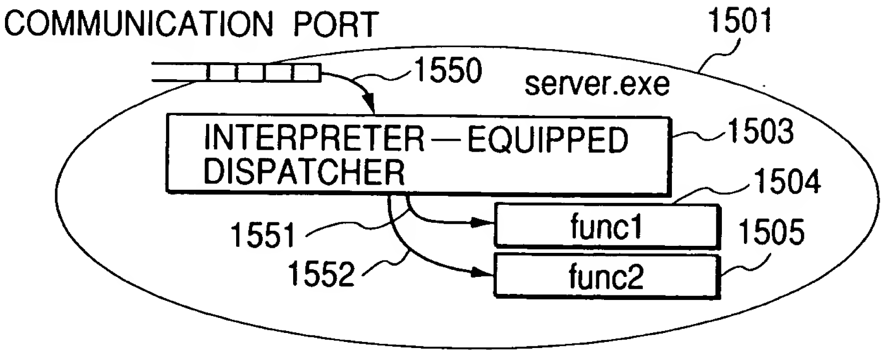

COMMUNICATION PORT
1501
1550
server.exe
INTERPRETER—EQUIPPED DISPATCHER
1503
1504
func1
1505
1551
1552
func2

FIG. 16

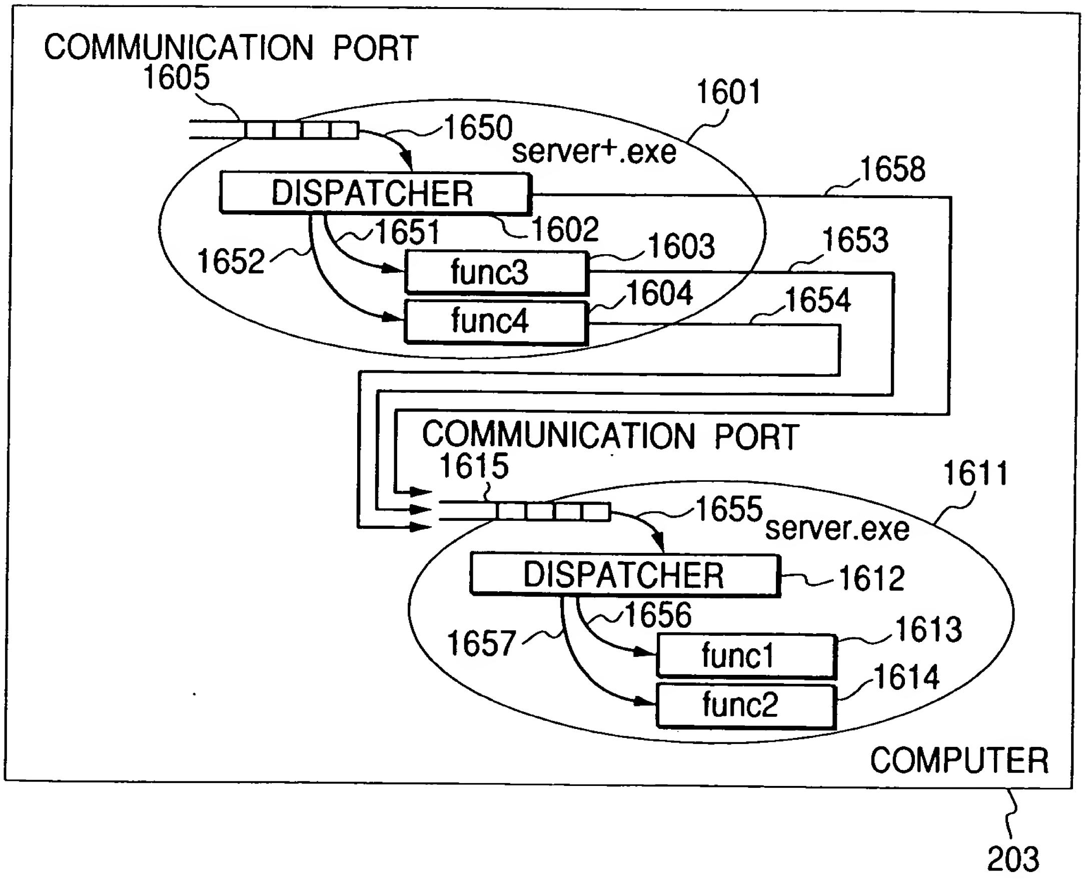

COMMUNICATION PORT
1605
1601
1650
server+.exe
DISPATCHER
1658
1651
1602
1603
1653
1652
func3
func4
1604
1654
COMMUNICATION PORT
1615
1611
1655
server.exe
DISPATCHER
1612
1656
1657
func1
1613
func2
1614
COMPUTER
203

FIG. 17

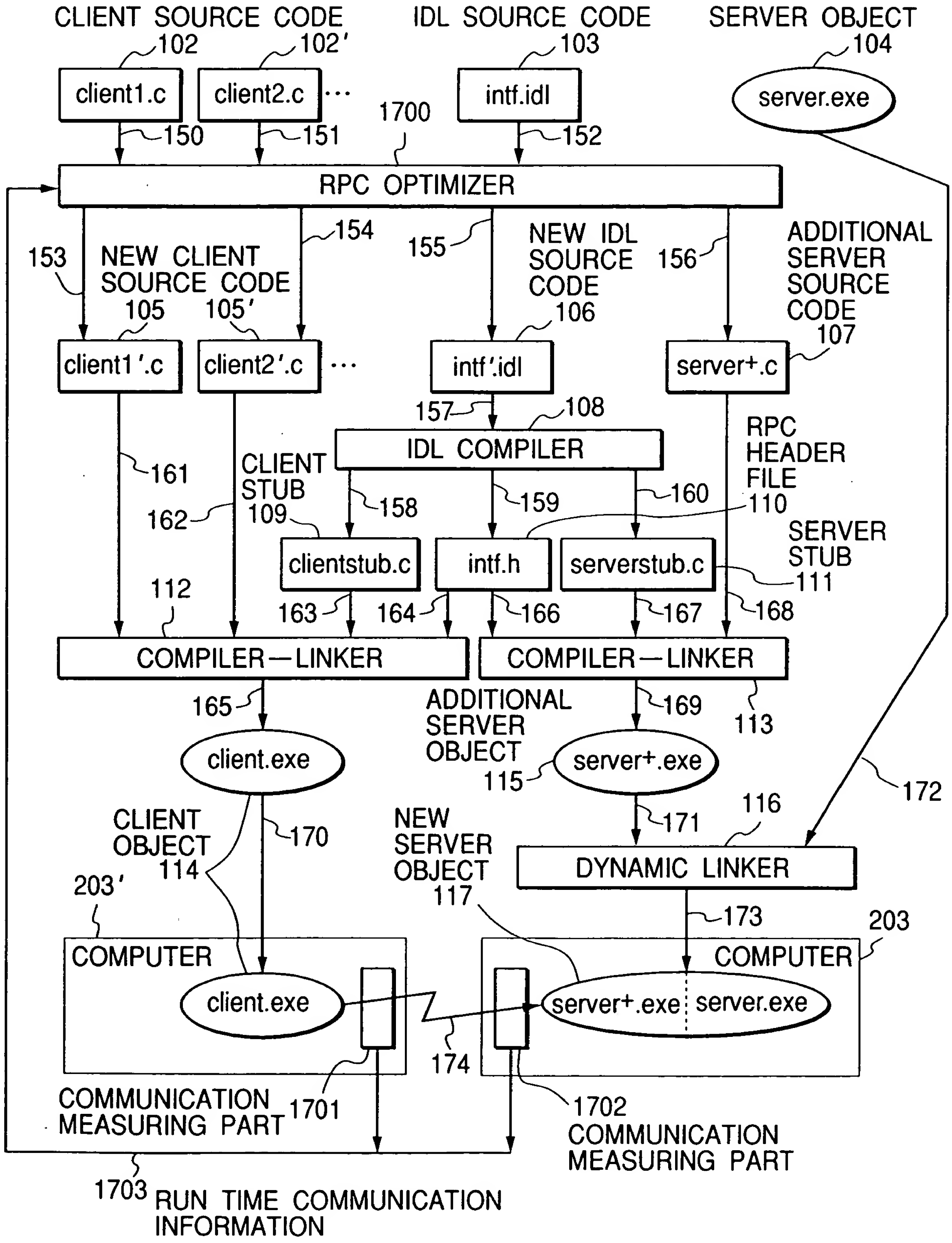

CLIENT SOURCE CODE
102
102'
client1.c
client2.c
IDL SOURCE CODE
103
intf.idl
SERVER OBJECT
104
server.exe
150
151
152
1700
RPC OPTIMIZER
153
NEW CLIENT SOURCE CODE
105
105'
154
155
NEW IDL SOURCE CODE
106
156
ADDITIONAL SERVER SOURCE CODE
107
client1'.c
client2'.c
intf'.idl
server+.c
157
108
IDL COMPILER
161
CLIENT STUB
109
162
158
159
160
RPC HEADER FILE
110
SERVER STUB
111
clientstub.c
intf.h
serverstub.c
112
163
164
166
167
168
COMPILER—LINKER
COMPILER—LINKER
165
169
113
client.exe
ADDITIONAL SERVER OBJECT
115
server+.exe
CLIENT OBJECT
114
170
NEW SERVER OBJECT
117
171
116
172
DYNAMIC LINKER
203'
173
203
COMPUTER
client.exe
COMPUTER
server+.exe
server.exe
174
COMMUNICATION MEASURING PART
1701
1702
COMMUNICATION MEASURING PART
1703
RUN TIME COMMUNICATION INFORMATION

FIG. 18

1800

extended intf.idl

```
1801 interface MyServer {
1802     int func1(in int i) const;
1803     void func2(inout long key, in String value);
1804     int func3(void);

1805     commutative { func2, func3 };

1806     parallel { func1, func2, func3 };
1807 };
```

1820

server+.c

```
1821 #include  "intf'.h"
1822 #include  "thread.h"

1823 void MyServer::func3(int& count)
1824 {
1825     List<Thread> allThreads;
1826     Thread t;
1827     void *rval;
1828     for (int i = 0; i < 100; i++) }
1829         create_thread(&t, server.func1, 1, i),;
1830          allThreads.add(t);
1831     }
1832     for ( ; (t = allThreads.next ()) != NULL_THREAD; ) {
1833         join_thread(t, &rval);
1834          count += *(int *)rval;
1835     }
1836 }

1837 void MyServer::func4(int count)
1838 {
1839     List<Thread> allThreads;
1840     Thread t;
1841         create_thread(&t, server.func2, 2, 100, "hello world");
1842          allThreads.add(t);
1843         create_thread(&t, server.func1, 1, count);
1844          allThreads.add(t);
1845          for ( ; (t = allThreads.next()) != NULL_THREAD; )
1846            join_thread(t, NULL);
1847 }
```

FIG. 19

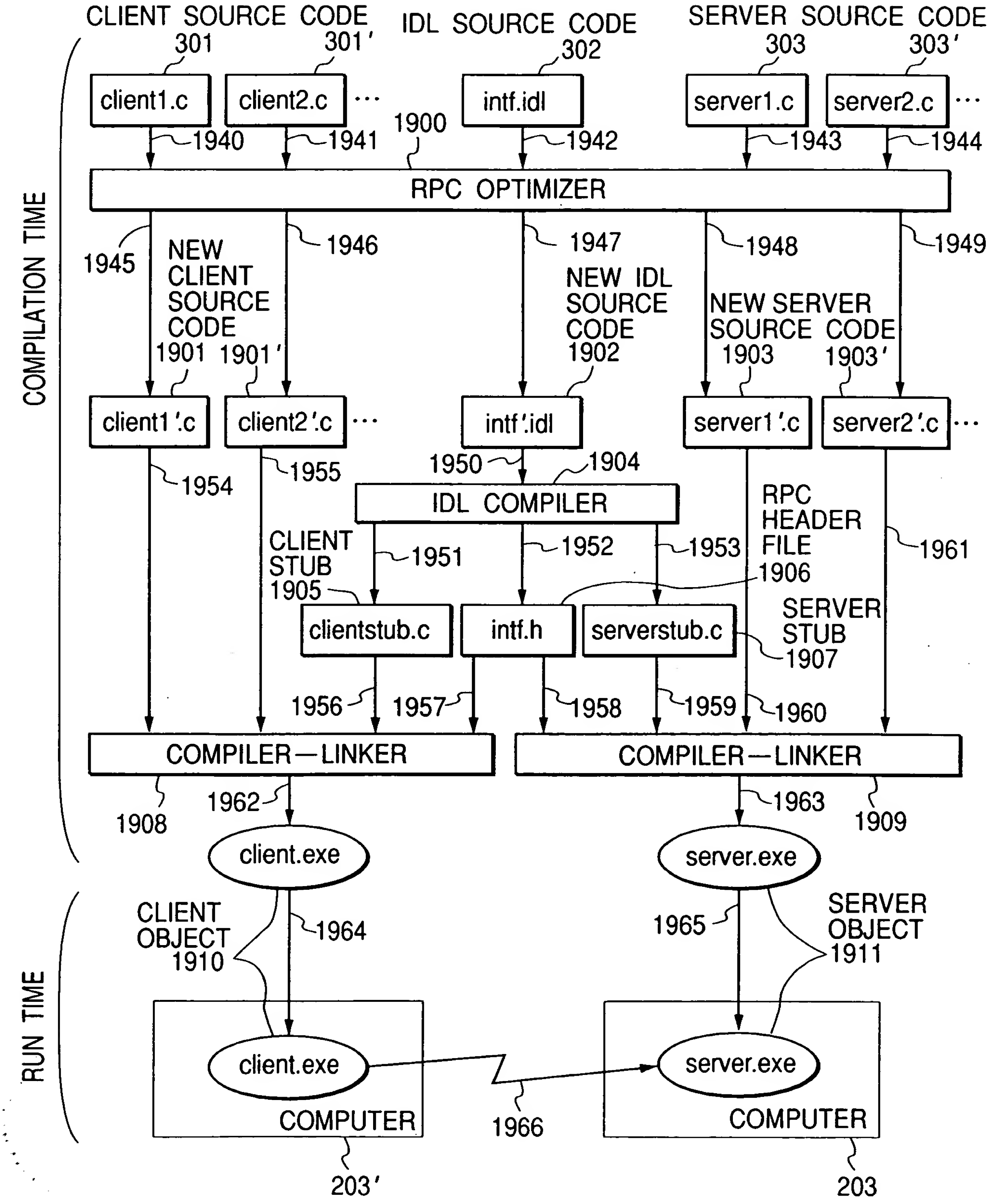

CLIENT SOURCE CODE
301
301'
IDL SOURCE CODE
302
SERVER SOURCE CODE
303
303'
client1.c
client2.c
intf.idl
server1.c
server2.c
1900
1940
1941
1942
1943
1944
RPC OPTIMIZER
COMPILATION TIME
1945
NEW CLIENT SOURCE CODE
1901
1901'
1946
1947
NEW IDL SOURCE CODE
1902
1948
NEW SERVER SOURCE CODE
1903
1903'
1949
client1'.c
client2'.c
intf'.idl
server1'.c
server2'.c
1954
1955
1950
1904
IDL COMPILER
CLIENT STUB
1905
1951
1952
1953
RPC HEADER FILE
1906
1961
clientstub.c
intf.h
serverstub.c
SERVER STUB
1907
1956
1957
1958
1959
1960
COMPILER—LINKER
COMPILER—LINKER
1908
1962
1963
1909
client.exe
server.exe
RUN TIME
CLIENT OBJECT
1910
1964
1965
SERVER OBJECT
1911
client.exe
server.exe
COMPUTER
COMPUTER
1966
203'
203

FIG. 20

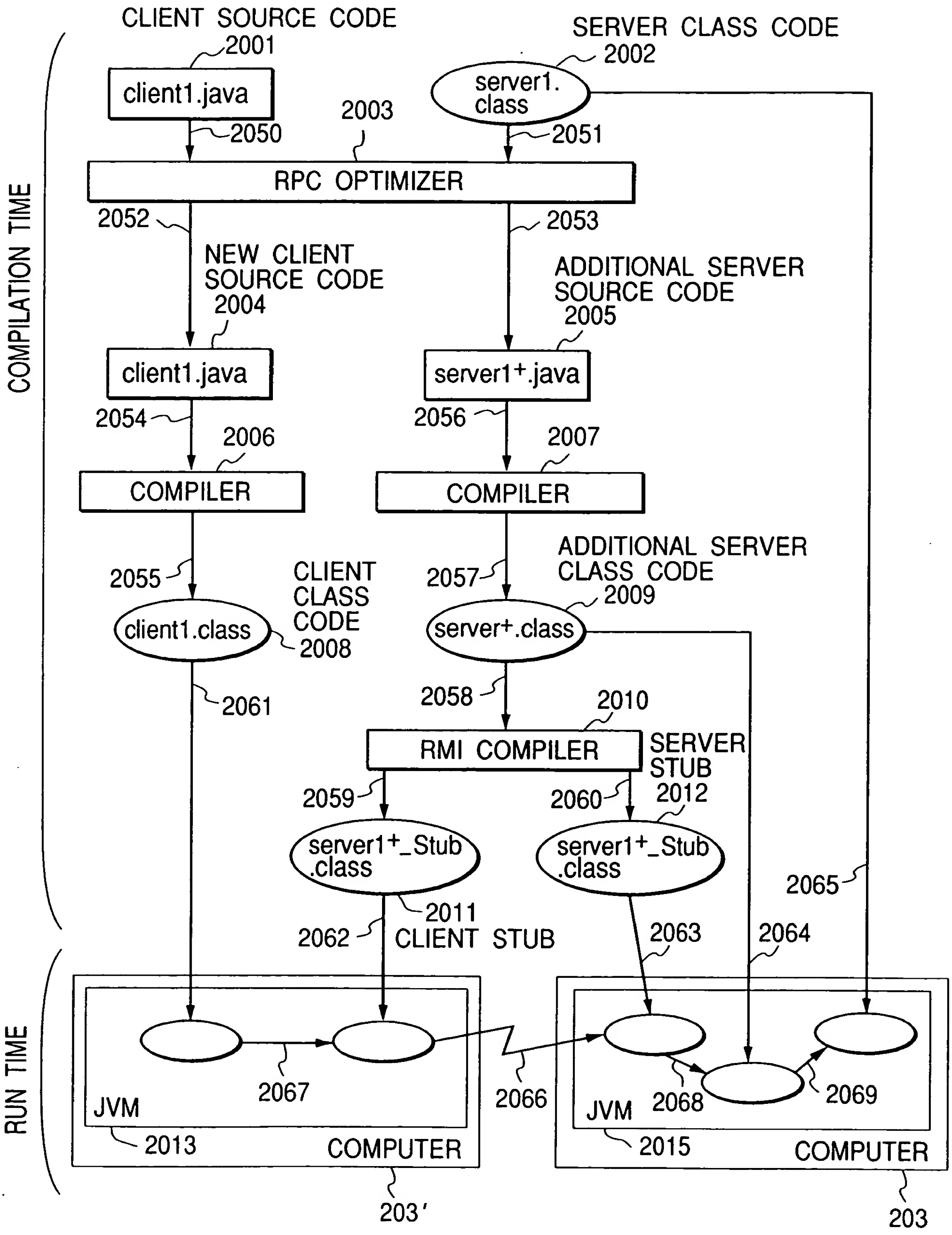
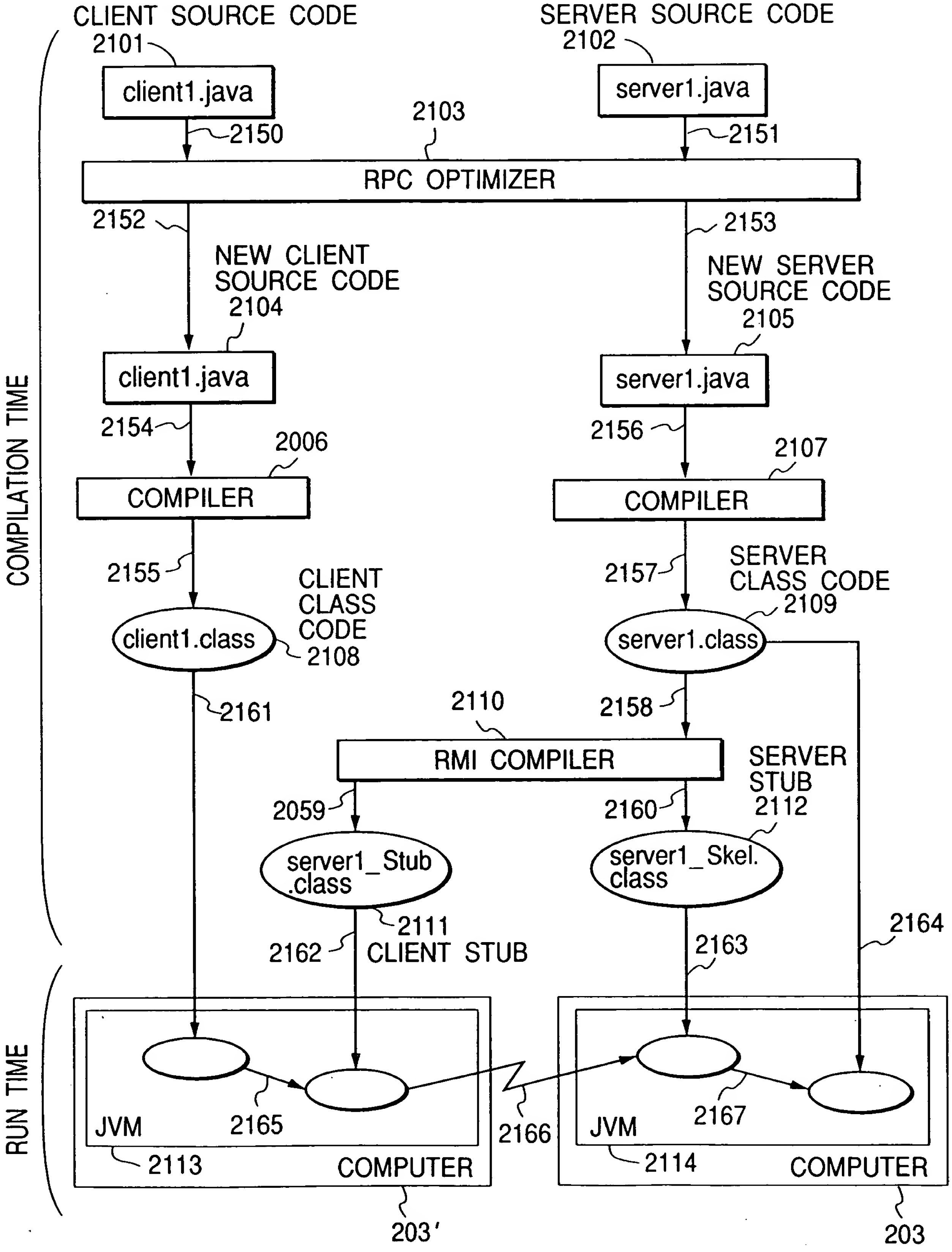

CLIENT SOURCE CODE
2001
client1.java
SERVER CLASS CODE
2002
server1. class
2003
2050
2051
RPC OPTIMIZER
2052
2053
NEW CLIENT SOURCE CODE
2004
ADDITIONAL SERVER SOURCE CODE
2005
client1.java
server1+.java
2054
2056
2006
2007
COMPILER
COMPILER
COMPILATION TIME
2055
CLIENT CLASS CODE
2008
client1.class
2057
ADDITIONAL SERVER CLASS CODE
2009
server+.class
2061
2058
2010
RMI COMPILER
SERVER STUB
2012
2059
2060
server1+_Stub .class
server1+_Stub .class
2065
2011
CLIENT STUB
2062
2063
2064
RUN TIME
2067
JVM
2013
COMPUTER
2066
JVM
2068
2069
2015
COMPUTER
203'
203

FIG. 21


CLIENT SOURCE CODE
2101
client1.java
SERVER SOURCE CODE
2102
server1.java
2150
2151
2103
RPC OPTIMIZER
2152
2153
NEW CLIENT SOURCE CODE
2104
NEW SERVER SOURCE CODE
2105
client1.java
server1.java
2154
2156
2006
2107
COMPILER
COMPILER
COMPILATION TIME
2155
2157
CLIENT CLASS CODE
2108
SERVER CLASS CODE
2109
client1.class
server1.class
2161
2110
2158
RMI COMPILER
SERVER STUB
2112
2059
2160
server1_Stub .class
server1_Skel. class
2111
CLIENT STUB
2162
2163
2164
RUN TIME
JVM
2165
2166
JVM
2167
2113
COMPUTER
2114
COMPUTER
203'
203